Decentralized, Distributed Internet Data Management

Abstract

A light-weight architecture is provided, where each component is in itself its own advanced mini-transaction processing monitor. To accomplish this, the system is most readily implemented as a set of Java classes. The resulting architecture is as follows. In a composite system, each server is an independent component performing its own scheduling and transaction management. These servers are built using Java and inheriting from the classes provided by the system according to the invention. The interface to each server defines the services it implements. An invocation of one of these services (through remote method invocation) results in the creation of a local transaction (child of the invoking transaction and parent of any transaction that might be triggered by invoking the services of other servers). Each transaction is a thread that (in an exemplary system) can invoke SQL statements in a local database (directly connected to that server) as well as services offered by other servers. All the information that is required to build a global composite transaction is implicitly added

by the system to each call. Each transaction is, however, independently handled at each server. That is, the servers neither communicate among themselves nor rely on a centralized component to make scheduling or recovery decisions. In this way, components can be dynamically added or removed from the system without compromising correctness. All a new server needs to know is the interface and address of the servers it will invoke. Regardless of the configuration, the system according to the invention guarantees that transactions executed over these servers will be correct (serializable) and recoverable at a global and local level.